07-31-00

A

FORM PTO-1082

Case Docket No.CA990022US1
0055.0022
July 28, 2000
Express Mail Label No. EL484106450US

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Dear Sir:

Transmitted herewith for filing is the patent application of

Inventor(s): S. S. Lightstone; C. S. McArthur; W. T. Oconnell; and M. A. Plasza
For: **HEURISTIC-BASED CONDITIONAL DATA INDEXING**

Enclosed are:

X 2 No. of Sheets of Drawings Sheet(s) of drawings ( X informal) + 0 extra copies;
29 pages of Application; 16 pages of specification, 1 page of abstract
___ An assignment of the invention to International Business Machines Corporation . (x Will follow.)
___ An associate power of attorney.
___ A verified statement to establish small entity status under 37 CFR 1.9 and 1.27.
X Unsigned Declaration and Power of Attorney. ( ___ Will follow.)
X Certified copy of Canadian Patent Application No. 2,279,119 filed July 29, 1999 from which priority is claimed under 35 U.S.C. §119.
___ IDS enclosed. ___ with references.

| CALCULATION OF FEES | | | | | | |
|---|---|---|---|---|---|---|
| | ITEM | NO. OF CLAIMS FILED MINUS BASE* | | NO. OF CLAIMS OVER BASE | X SM/LG ENTITY FEE | $ AMOUNT | $ FEE |
| A | TOTAL CLAIMS FEE | 56 | - 20* = | 36 | X $9 or $18 | $648 | |
| B | INDEPENDENT CLAIMS FEE** | 8 | - 3* = | 5 | X $39 or $78 | $390 | |
| C | SUBTOTAL - ADDITIONAL CLAIMS FEE (ADD FINAL COLUMN IN LINES A + B) | | | | | | 1,038 |
| D | MULTIPLE-DEPENDENT CLAIMS FEE | SMALL ENTITY FEE = $130; LARGE ENTITY FEE = $260 | | | | | $0 |
| E | BASIC FEE* | SMALL ENTITY FEE = $345; LARGE ENTITY FEE = $690 | | | | | $690 |
| F | TOTAL FILING FEE (ADD TOTALS FOR LINES C, D, AND E) | | | | | | $1,728 |
| G | ASSIGNMENT RECORDING FEE | | | | | $ 40 | $0 |
| | **LIST INDEPENDENT CLAIMS 1, 12, 13, 18, 20, 27, 37, 47 | | | | | | |

Please charge Dep. Acct. No. 50-0585 in the amount of $ **A copy of this sheet is enclosed.**

X A check in the amount of $ 1,728 to cover the filing fee is enclosed.
___ Check for $___ covering the Recordation of Assignment fee enclosed.
X The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 50-0585 . **A copy of this sheet is enclosed.**
X Any additional filing fees required under 37 CFR 1.16.
X Any patent application processing fees under 37 CFR 1.17.
___ The Commissioner is hereby authorized to charge payment of the following fees during the pendency of this application or credit any overpayment to Deposit Account No. 50-0585. **A copy of this sheet is enclosed.**
___ Any patent application processing fees under 37 CFR 1.17.
___ The issue fee set in 37 CFR 1.18 at or before mailing of the Notice of Allowance, pursuant to 37 CFR 1.311(b).
___ Any filing fees under 37 CFR 1.16 for presentation of extra claims.

24033
PATENT TRADEMARK OFFICE

Respectfully submitted,

David W. Victor
Registration No. 39,867

Direct All Correspondence to:
David W. Victor
KONRAD RAYNES & VICTOR LLP
1180 S. Beverly Drive; Suite 501
Los Angeles, CA 90035

Direct Telephone Calls to:

(310) 556-7983

Docket No. CA990022US1
Firm No. 0055.0022

# HEURISTIC-BASED CONDITIONAL DATA INDEXING

## RELATED FOREIGN APPLICATION

This patent application claims priority from the commonly assigned Canadian Patent Application entitled "Heuristic-Based Conditional Data Indexing", having Canadian Patent Application Serial No. 2,279,119, filed on July 29, 1999 by Sam S. Lightstone, Miroslaw A. Flasza, Catherine S. McArthur and William T. O'Connell, which application is incorporated herein by reference in its entirety.

## 1. Field of the Invention

The present invention is directed to an improvement in computing systems and in particular to computer systems which include the indexing of data.

## 2. Background of the Art

Computer systems for the storage and retrieval of data typically include indexes of the data which are created to permit operations on the data (data retrieval or updating) to be performed in an efficient manner. Updating such indexes is typically costly in system resources and time. For example, in a relational database environment, appending a new record to a table in the database will require a new entry in any indexes which have been created for that table. The time and computer resources required to add a set of data to the table may be significantly less than the operation of updating the associated indexes.

In database management system (DBMS) environments, it is typically the case that the efficiency or inefficiency of the index updating operations significantly impacts on the efficiency of the database operations as a whole.

It is therefore desirable to have a DBMS computer system which includes features to make more efficient the updating of indexes associated with data stored by the DBMS.

## SUMMARY OF THE PREFERRED EMBODIMENTS

According to one aspect of the preferred embodiments, there is provided an improved system for indexing data.

According to another aspect of the preferred embodiments, there is provided an improved method, system, and program for updating an index on a database table when data is added to the table. When data records are received to load into the table, a selection is made of a first operation or second operation. The first operation incrementally updates the index on the table as each record in the received data is added to the table and the second operation rebuilds the index from the table after all the received data records have been added to the table. The selected first or second operation is used to update the index with the received data.

In still further embodiments, a determination is made as to which of the first operation or second operation is more efficient. The first or second operation determined to be more efficient is the selected operation used for updating the index with the received data.

According to another aspect of the preferred embodiments, there is provided a computer system for database management comprising, means for storing and updating a first set of data, indexing means for storing and updating a selected index of key values related to the first set of data, the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, heuristic determination means for selecting the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data.

According to another aspect of the preferred embodiments, there is provided the above computer system in which the heuristic determination means comprises a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

According to another aspect of the preferred embodiments, there is provided the above computer system in which the database management system is a relational database

management system in which the index is stored as a binary tree and in which the index meta-data comprises estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index.

According to another aspect of the preferred embodiments, there is provided the above computer system in which the function determines, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

According to another aspect of the preferred embodiments, there is provided the above computer system in which the heuristic determination means comprises means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

According to another aspect of the preferred embodiments, there is provided the above computer system further comprising means for generating values for the function of the heuristic determination means, for a specified test range of values for each of the first set of data, the index, and the second set of data.

According to another aspect of the preferred embodiments, there is provided the above computer system, the computer system having one or more CPUs, one or more disks, a sort heap and a database bufferpool, and in which the index meta-data is defined to reflect a subset of the following characteristics: the percentage of free space in the index, the estimated size of the index after both the incremental and the rebuild updates of the index, the width of the average key value in the index, the size of the sort heap and the database bufferpool in the computer system, the number and speed of the CPUs in the computer system, and the number and speed of the disks in the system.

According to another aspect of the preferred embodiments, there is provided a computer system for relational database management comprising, means for storing and updating a first set of data, indexing means for storing as a binary tree, and updating, a selected index of key values related to the first set of data the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, heuristic determination means for selecting the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, the heuristic determination means comprising a function which takes as input index meta-data, the index meta-data being stored separately from the first set of data and comprising estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index, the heuristic determination means further comprising means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

According to another aspect of the preferred embodiments, there is provided a computer program product for use with a computer comprising a central processing unit and random access memory, said computer program product comprising a computer usable medium having computer readable code means embodied in said medium for managing a database, said computer program product comprising computer readable program code means for causing a computer to implement the system functionality described above.

According to another aspect of the preferred embodiments, there is provided a method for updating a selected index in a computer system for database management, the computer system comprising means for storing and updating a first set of data related to the index, the method comprising the steps of:

(a) heuristically selecting one of: the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, and

(b) selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, as selected.

According to another aspect of the preferred embodiments, there is provided the above method in which the step of heuristically selecting the update of the index comprises the evaluation of a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

According to another aspect of the preferred embodiments, there is provided a method for updating a selected b-tree index in a computer system for relational database management, the computer system comprising means for storing and updating a first set of data related to the index, the method comprising the steps of:

(a) using a heuristic determination function to select one of: the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, the heuristic determination function taking as input index meta-data comprising estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the heuristic function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index, and

(b) selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, as selected.

According to another aspect of the present invention, there is provided the above method in which the function comprises the step of determining, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

According to another aspect of the present invention, there is provided the above method further comprising the step of a user specifying the selection of the incremental update of the index, the full rebuild of the index, or the selection based on the heuristic determination function.

According to another aspect of the present invention, there is provided a computer program product tangibly embodying a program of instructions executable by a computer to perform the above method steps.

Advantages of the present invention include improvements in the efficiency in updating a database when loading data or when rolling forward data after a database restore.

## BRIEF DESCRIPTION OF THE DRAWINGS

The preferred embodiment of the invention is shown in the drawings, wherein:

Figure 1 is a block diagram showing the data structures used by the preferred embodiment in a data roll forward.

Figure 2 is a block diagram showing the data structures used by the preferred embodiment prior to data being loaded into a database in the preferred embodiment.

Figure 3 is a block diagram showing the data structures used by the preferred embodiment during data being loaded into a database in the preferred embodiment.

In the drawings, the preferred embodiment of the invention is illustrated by way of example. It is to be expressly understood that the description and drawings are only for the purpose of illustration and as an aid to understanding, and are not intended as a definition of the limits of the invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The preferred embodiment is described with reference to the DB2 database management system (DBMS).** Those skilled in the art will appreciate that the preferred embodiment may be implemented in other DBMS environments other than DB2. The preferred embodiment of the invention relies on the fact that for some database operations

a series of transactions will result in data being updated without there being a need to access indexes associated with the data.

For example, a DBMS may permit a user to load data into a table in the database from another source (for example where data is copied from a first database into a second database). The DB2 LOAD command is an example of this sort of operation. The LOAD command will result in data being inserted into a table. It is possible for large data collections to be added to tables using the LOAD command. In one type of prior art system, the DBMS will incrementally update indexes associated with the table into which data is placed using the LOAD command. In such a system, the index on the table will be modified on a record by record basis to reflect the new records added to the table by the LOAD command. However, the index will not be used, in the sense of being accessed to locate data in the database, until the LOAD command has been completed and all the data has been added to the target table in the database. In other systems, the index will not be updated in an incremental fashion but will be fully rebuilt following the addition of new records to the data in the database.

Another example of where a DBMS adds to or modifies data without the use of an index until the changes to the data are complete is in the ROLLFORWARD command in DB2. When data in a relational database is recovered by using the RESTORE and ROLLFORWARD commands, the DBMS facilitates the rebuilding of a database from a transaction log where the database has been destroyed or become corrupted in some manner (a database "roll forward"). During the roll forward process, the data in the database is modified to reflect the transactions stored in the transaction log. In the prior art, it is common to update associated indexes on a record by record basis which reflects the transactions stored in the transaction log. For DBMS environments such as the DB2 system, the transaction log stores a portion of the transactional history which permits the updating of the data in the database after a database restore, without requiring reference to the indexes associated with the data being updated or rolled forward. The indexes associated with the data must be brought up to date at the end of the roll forward operation, but need not be synchronized with the data during the operation.

As indicated above, for certain DBMS environments, the DBMS will update indexes on a transaction by transaction or record by record basis. For certain DBMS environments, the content of the indexes during the data updating process (during the LOAD or the roll forward command execution) is not accessed by the DBMS. The DBMS requires a correct set of indexes at the completion of the operations but is not concerned with the intermediate values of the indexes.

As the above indicates, in the prior art, one approach in loading data or performing a roll forward on a database, is to update the database on a record by record basis. In such a system any indexes which are associated with the data being updated will also be modified on a record by record basis. Alternatively, certain DBMS designs will permit indexes to be rebuilt at the end of a roll forward operation. In certain DBMS designs, it is also possible for the system itself to select an index rebuild in certain circumstances where otherwise a record by record or incremental update of the index would occur. In the DB2 DBMS, for example, the default approach during the ROLLFORWARD command execution is the incremental updating of the index. The index rebuilding operation is only carried out where one of the transactions being replayed in a roll forward is an index create function. Otherwise, the entire set of transactions is replayed on the index.

Where data is loaded or a database is subject to a roll forward operation, it is possible that a significant proportion of the processing time for the operation will be taken up by the addition to or modification of the indexes associated with the data. In fact, for certain databases it may be more efficient to ignore the incremental updating of the associated indexes and to fully rebuild the indexes at the completion of the loading of data or the roll forward operations.

Turning to Figure 1, an example database 10 is illustrated. Database 10 contains table 12 and index 14. For the purposes of the description of the preferred embodiment a relational database is described. It will be apparent to those skilled in the art that other database systems may be used to implement the preferred embodiment, with the appropriate changes to accommodate differences from the relational database model. In

the preferred embodiment, there is a collection of meta-data shown in Figure 1 as meta-data 16.

In the preferred embodiment, a backup of the database is made at periodic or user-defined intervals. In Figure 1, backup database 18 is shown with table 20 and index 22. Backup database 18 represents stored data for database 10. In the system of the preferred embodiment, there is also a set of transactions stored in log 24. The stored transactions are shown as the set $Tx_0$, $Tx_1$, $Tx_2$,...$Tx_n$. In the preferred embodiment, the set of transactions stored in log 24 are used by the system in conjunction with backup database 18 to recover data which may have been lost or inadvertently deleted. In the system of the preferred embodiment, the database 10 may be replaced with the data from backup database 18 (using the RESTORE command in the DB2 DBMS, for example). Log 24 contains transactions which have occurred since the time that backup database 18 was created. The stored transactions in log 24 are able to be replayed on recovered data in database 10, to update table 12 and index 14 in database 10. One prior art approach is to update index 14 as each transaction in log 24 is replayed on the data in database 10. Index 14 is therefore updated on a record by record (or transaction by transaction, basis). This is the approach used by the roll forward command available in the DB2 DBMS.

According to the preferred embodiment, the DBMS includes functionality to recover data using commands such as RESTORE and ROLLFOWARD, which does not necessarily require the record by record updating of index 14. In the preferred embodiment, backup database 18 may be used to replace the table 12 in database 10. However, when the transactions in log 24 are replayed on database 10 as in a roll forward, the system of the preferred embodiment will selectively permit table 12 to be updated, without updating index 14 (as is set out above, log 24 contains data for each of the stored transactions to permit updating of table 12 without reference to index 14). As the time for incrementally updating index 14 is potentially long, the savings in updating table 12, only are potentially significant. Once table 12 has been updated by the transactions in log 24, index 14 will be recreated by being rebuilt on the newly updated table 12.

Such a method of updating by ignoring index 14 until the completion of updating table 12 (the full index rebuild) is available in the preferred embodiment by user selection. Alternatively, the full index rebuild roll forward is available by the system of the preferred embodiment applying a heuristic determination function to meta-data 16.

Index meta-data 16 is represented as $S_i$. A heuristic determination function $F(S_i)$ is defined in the system of the preferred embodiment to accept the set of heuristic information and to determine whether indexing in a roll forward should be maintained transactionally (incremental updating), or deferred for a single recreation pass (full index rebuild).

The determination function $F(S_i)$ may accept as parameters such database meta-data as the estimated number of key-parts assumed to comprise an index at the time roll forward processing starts, and an estimate of the number of keypart updates that will be applied during roll forward processing. The specific decision making logic of the heuristic determination function $F(S_i)$ may be dependent on implementation details, for specific DBMS systems. The $F(S_i)$ definition may have considerable empirical dependence, which may vary between products, and between product releases.

A simple implementation of function $F(S_i)$ has parameters reflecting characteristics of an index 22, and compares these to a set of index characteristics at roll forward completion time which have been maintained in meta-data 16. In the specific implementation for the DB2 Universal Database, the preferred embodiment stores the number of index key operations as meta-data inside the index object meta-index. This information is periodically reflected in the database recovery history file during normal transaction processing of the DBMS. In the preferred embodiment this information is stored outside the database to maintain the information should the database crash. During roll forward recovery processing the desired heuristic information reflecting end-of-logs can be retrieved from the Recovery History File, and provided to the heuristic determination function $F(S_i)$.

In the system of the preferred embodiment, the heuristic determination function $F(S_i)$ is used also when pre-existing data is loaded into a database. Figure 2 shows

database 10 with table 12 and index 14. Data 26 is data to be added to database 10. In DBMS systems such as DB2, commands for loading data such as data 26 into database 10 are typically available (the LOAD command in DB2). In the system of the preferred embodiment, the data may be loaded such that index 14 is incrementally updated, or the data may be added to table 12 in database 10 without incrementally updating index 14. At the end of adding data 26 to table 12, index 14 may then be recreated using the updated data in table 12.

The implementation of the loading of data in the preferred embodiment system may be described with reference to Figure 3. The first step in loading data 26 is the addition of the data to table 12. The added data 38 shown in table 12 in Figure 3, is added from data 26 (shown in Figure 2). The system of the preferred embodiment also creates a data structure to hold the sorted keys 30 f or added data 38. The sorted keys 30 relate to the keys in index 14. If the user or the heuristic determination function F(Si) selects an incremental update of index 14, then sorted keys 30 are added to index 14, on a key value by key value basis. Alternatively, if the user or function $F(S_i)$ determines that index 14 should be recreated, then index 14 will be rebuilt on the entire table 12 key values.

As is apparent, in both the data recovery and roll forward operations, and in the data load operation, the use of the heuristic determination function $F(S_i)$ is available in the system of the preferred embodiment. The function will be used to select incremental updating index 14, or alternatively to select a full rebuilding of index 14. In the system of the preferred embodiment, the user may either manually select incremental or full rebuilding modes or may select a system option to have the system itself use the heuristic determination function $F(S_i)$ to make the selection.

In both the load and restore/roll forward operations, the heuristic determination function $F(S_i)$ is intended to predict which of the two index updating approaches (incremental and full rebuild) will be most efficient for the database updating which is being performed. The case where database 10 is relatively large, and few data values are added to database 10 (and correspondingly to index 14), will be, for many DBMS

environments, one where incremental updating of index 14 will be more efficient. Conversely, where database 10 is relatively small and the data to be added by either a large log 24 or a large set of data 26, in many DBMS environments, it will be more efficient to update index 14 by fully rebuilding the index after table 12 has been updated. Typically, it is the database subsystem and environment which will determine which of the two different updating methods is more efficient, for a given data and index updating operation. The heuristic determination function $F(S_i)$ is intended to predict which of the two methods is more efficient, given a particular combination of existing database, update data, database subsystem and environment. Once the appropriate values for the function are set for a given database subsystem and environment, it is intended that the same function $F(S_i)$ will be appropriate for different table and index combinations in different database definitions.

In the system of the preferred embodiment, index 14 is stored in the form of a binary tree, such as a B-tree, B+-tree, or B*-tree. It is to be understood, however, that other index structures may be used and the benefit of the invention achieved. The preferred embodiment uses two characteristics of a binary tree index to model the efficiency of incremental updating and full rebuilding for an index. The two characteristics are the percentage of new data that is to be added to table 12 and the height of the binary tree structure of index 14. The tree height of index 14 is a measure of the size and complexity of index 14. In the example of the preferred embodiment, there is a single index shown. In the system of the preferred embodiment, the first table non-meta-index table index is taken to be representative of the database indexes which will be affected by the updating procedure. It will be understood by those skilled in the art that there may be other approaches to selecting an index which is representative of the indexes in a database for which data is to be updated. The approach of selecting the first such index and assuming that that index is representative has the advantage of carrying low storage and processing overheads. It may be advantageous in certain circumstances to maintain other data regarding the indexes such as the largest index, the smallest index, or the median or average index. Such approaches may require more system resources to

maintain but will have the advantage of potentially providing better information about the structure of the indexes for the determination heuristic $F(S_i)$.

In defining $F(S_i)$ in terms of tree height of index 14, and the percentage of new data to be added to table 12, $F(S_i)$ is designed to reflect the time required to extract index keys from table 12. This operation is necessary in a full rebuild of index 14 and requires a scan of all records in table 12. The cost of this is generally linear with the size of the data on disk. Typically, each page must be scanned from disk, then scanned for records, and each record scanned for index keys.

The function $F(S_i)$ is also designed to reflect the time to sort index keys. This varies with the sorting algorithm used, the size of buffering for sort operations, and the cost of spilling to disk (if required). Generally, however, this cost is modelled as an NlogN operation.

The function is also intended to reflect the time to fully rebuild index 14 from a set of sorted keys. This is generally a linear operation, which varies with the number of key values to be included in index 14. Conversely, the function will be used to reflect the time to incrementally extend index 14 with a set of sorted keys. This is generally a non-linear operation, which is a function of the height of the binary index tree.

In the model for $F(S_i)$ of the preferred embodiment, a threshold value for percentage of new data is determined for different tree heights. The threshold value will indicate at what point it is estimated that incremental updating of index 14 is to be used (the threshold value and below) and at what point a full rebuild approach to index 14 should be used (above the threshold value).

In the preferred embodiment, the $F(S_i)$ is essentially a table:

| Tree Height | Threshold |
|---|---|
| 2 (or less) | T2 |
| 3 | T3 |
| 4 | T4 |
| 5 | T5 |
| 6 | T6 |
| 7 | T7 |
| 8 | T8 |

In the table T2...T8 represent the percentile thresholds for the tree height 2 to 8 for which incremental indexing proves to yield superior performance for a given DBMS. As the tree height grows, incremental indexing threshold values will vary. The incremental costs of sorting data need to be balanced against the cost of traversing the tree. Since both sorting and scanning the binary tree vary as NlogN operations, to some extent the way these two factors trade off with varying amounts of new and old data depends on the implementation of the key sorting and tree insertion logic. Due to the dependency on implementation, the preferred embodiment models these costs empirically. Tests are performed to examine at what volumes of new index key values, incremental indexing performed better than full index rebuilding.

In the preferred embodiment, the $F(S_i)$ for the DB2 product was determined by using as benchmark data a set of data provided by the Transactional Processing Performance Counsel. Based on that benchmark data, the values for the thresholds for tree heights 2 to 8, inclusive, for the DB2 product were calculated as:

T2 = 16

T3 = 19

T4 = 21

T5 = 24

T6 = 27

T7 = 30

T8 = 33

Additional values may be calculated for tree heights greater than eight, as required. The above values indicate that where index 14 has a height 6, for example, and the data to be added to table 12 is above 27% of the size of table 12 (the number of records to be added is more than 27% of the number of records in table 12), then a full rebuild of index 12 will be more efficient than adding key values to index 12 incrementally. It is to be understood that the above values may vary for different DBMS environments and data sets.

Although the above table values for the function $F(S_i)$ were arrived at by taking test data and manually constructing indexes to determine the thresholds for the different heights of trees, such a process need not be carried out by a user. Once the location of the test data is indicated to the system of the preferred embodiment, it is possible for the system to run the tests to come up with the appropriate threshold values as part of the system set up. It is necessary for the system to clock its own processes and to select the appropriate threshold points based on the data provided.

For indexes such as hash indexes which are not based on a binary tree structure, the heuristic determination function may compare percentages of new data to total data volume to achieve a heuristic to predict when incremental or full rebuild approaches are more efficient.

Other characteristics of database 10 which may be included in determining the function $F(S_i)$ include:

1. The percentage of freespace in index 14;

2. The estimated size of index 14 after either incremental or full rebuild updates (a full rebuild will often reduce fragmentation);

3. The width of the average index key;

4. The size of the sort heap and the database bufferpool (for memory caching);

5. The number and speed of the CPUs in the system; and

6. The number and speed of disks in the system.

Each of the above characteristics may have an impact on the relative efficiency of incremental and full rebuild updating of index 14. For that reason, inclusion of these values in the function $F(S_i)$ may provide additional predictive accuracy for the function.

Although a preferred embodiment of the present invention has been described here in detail, it will be appreciated by those skilled in the art, that variations may be made thereto, without departing from the spirit of the invention or the scope of the appended claims.

---

**DB2 is a registered trademark of International Business Machines Corp.

WHAT IS CLAIMED

1. A computer system for database management comprising, means for storing and updating a first set of data,

indexing means for storing and updating a selected index of key values related to the first set of data, the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index,

heuristic determination means for selecting the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data.

2. The computer system of claim 1 in which the heuristic determination means comprises a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

3. The computer system of claim 2 in which the database management system is a relational database management system in which the index is stored as a binary tree and in which the index meta-data comprises estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index.

4. The computer system of claim 3 in which the function determines, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

5. The computer system of claim 1 in which the heuristic determination means comprises means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

6. The computer system of claim 2 further comprising a means for storing the index meta-data independent of the means for storing and updating the first set of data.

7. The computer system of claim 2 in which the means for storing the index meta-data comprises a recovery history file for the first set of data.

8. The computer system of claim 1 in which the selected index of key values is one of a plurality of indexes and in which the selected index is the first index on the first set of data.

9. The computer system of claim 1 in which the selected index of key values is one of a plurality of indexes and in which the selected index is selected on the basis of the relative sizes of each of the plurality of indexes on the first set of data.

10. The computer system of claim 2 further comprising means for generating values for the function of the heuristic determination means, for a specified test range of values for each of the first set of data, the index, and the second set of data.

11. The computer system of claim 2, the computer system having one or more CPUs, one or more disks, a sort heap and a database bufferpool, and in which the index meta-data is defined to reflect a subset of the following characteristics: the percentage of free space in the index, the estimated size of the index after both the incremental and the rebuild updates of the index, the width of the average key value in the index, the size of the sort heap and the database bufferpool in the computer system, the number and speed of the CPUs in the computer system, and the number and speed of the disks in the system.

12. A computer system for relational database management comprising,

means for storing and updating a first set of data,

indexing means for storing as a binary tree, and updating, a selected index of key values related to the first set of data the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index,

heuristic determination means for selecting the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, the heuristic determination means comprising a function which takes as input index meta-data,

the index meta-data being stored separately from the first set of data and comprising estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index,

whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index,

the heuristic determination means furthercomprising means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

13. A computer program product for use with a computer comprising a central processing unit and random access memory, said computer program product comprising a computer usable medium having computer readable code means embodied in said medium for managing a database, said computer program product comprising:

computer readable program code means for causing a computer to store and update a first set of data,

computer readable program code indexing means for causing a computer to store and update a selected index of key values related to the first set of data, the indexing means comprising means for selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index,

computer readable program code heuristic determination means for causing a computer to select the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data.

14. The computer program product of claim 13 in which the heuristic determination means comprises a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

15. The computer program product of claim 14 in which the database is a relational database in which the index is stored as a binary tree and in which the index meta-data comprises estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index.

16. The computer program product of claim 15 in which the function determines, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data, whereby the threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

17. The computer program product of claim 13 in which the heuristic determination means comprises means for the user to specify the selection of the incremental update of the index or the full rebuild of the index.

18. A method for updating a selected index in a computer system for database management, the computer system comprising means for storing and updating a first set of data related to the index, the method comprising the steps of:

(a) heuristically selecting one of: the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, and

(b) selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, as selected.

19. The method of claim 18 in which the step of heuristically selecting the update of the index comprises the evaluation of a function which takes as input index meta-data, comprising characteristics of the first set of data, the index, and the second set of data.

20. A method for updating a selected b-tree index in a computer system for relational database management, the computer system comprising means for storing and updating a first set of data related to the index, the method comprising the steps of:

(a) using a heuristic determination function to select one of: the incremental update of the index, or the full rebuild of the index, for a given second set of data to be added to the first set of data, the heuristic determination function taking as input index meta-data comprising estimates of the table size of the first set of data, the table size of the second set of data, and the height of the index, whereby the heuristic function is empirically defined to select the incremental update of the index or the full rebuild of the index based on the predicted relative efficiencies of the incremental update of the index and the full rebuild of the index, and

(b) selectively updating the index by incrementally updating the index by incrementally adding key values to the index or by fully rebuilding the index, as selected.

21. The method of claim 20 in which the function comprises the step of determining, for each potential binary tree height of the index, a threshold percentage of the table size of the second set of data to the table size of the first set of data, whereby the

threshold percentage value for each binary tree height determines the selection of the incremental index update or the full rebuild index update.

22. The method of claim 18 further comprising the step of a user specifying the selection of the incremental update of the index, the full rebuild of the index, or the selection based on the heuristic determination function.

23. The method of claim 19 further comprising the step of storing the index meta-data independent of the means for storing and updating the first set of data.

24. A computer program product tangibly embodying a program of instructions executable by a computer to perform the method steps of claim 18.

25. A computer program product tangibly embodying a program of instructions executable by a computer to perform the method steps of claim 20.

26. A computer program product tangibly embodying a program of instructions executable by a computer to perform the method steps of claim 21.

27. A method for updating an index on a database table when data is added to the table, comprising:

receiving data records to load into the table;

selecting one of a first operation and second operation, wherein the first operation incrementally updates the index on the table as each received data record is added to the table and the second operation rebuilds the index from the table after all the received data records have been added to the table; and

using the selected first operation or second operation to update the index with the received data.

28. The method of claim 27, further comprising:

determining which of the first operation or second operation is more efficient, wherein the first or second operation determined to be more efficient is the selected operation used for updating the index with the received data.

29. The method of claim 28, wherein determining which operation is more efficient is a function of a percentage of the received data records to add to the table and characteristics of the index.

30. The method of claim 29, wherein the characteristics of the index used in determining which operation is more efficient comprise a size and complexity of the index.

31. The method of claim 30, wherein the index comprises a binary tree structure, and wherein a height of the index tree is indicative of the size and complexity of the index.

32. The method of claim 28, wherein determining which operation is more efficient further comprises considering at least one of a following factors: an estimated time required to extract index keys from the table; an estimated time to sort the index keys; and an estimated time to rebuild the index from the sorted keys.

33. The method of claim 28, further comprising:

maintaining a list of threshold values for different index sizes; and

using the number of received data records to add to the table to determine a comparison value, wherein determining whether the first or second operation is more efficient is based on the comparison value and the threshold for the size of the index to be updated.

34. The method of claim 33, wherein the comparison value comprises the number of the received data records as a percentage of all data records in the table.

35. The method of claim 34, wherein the index comprises a binary tree and wherein the list of threshold values provides one threshold for each of a plurality of different height index binary trees, wherein the threshold selected for comparison with the comparison value is based on the height of the index to update.

36. The method of claim 33, wherein the first operation is more efficient if the comparison value is less than the threshold value and wherein the second operation is more efficient if the comparison value is greater than the threshold value.

37. A system for updating an index on a database table when data is added to the table, comprising:

a database system including the table and the index on the table;

means for receiving data records to load into the table;

means for selecting one of a first operation and second operation, wherein the first operation incrementally updates the index on the table as each received data record is added to the table and the second operation rebuilds the index from the table after all the received data records have been added to the table; and

means for using the selected first operation or second operation to update the index with the received data.

38. The system of claim 37, further comprising:

means for determining which of the first operation or second operation is more efficient, wherein the first or second operation determined to be more efficient is selected to use for updating the index with the received data.

39. The system of claim 38, wherein the means for determining which operation is more efficient is a function of a percentage of the received data records to add to the table and characteristics of the index.

40. The system of claim 39, wherein the characteristics of the index used in determining which operation is more efficient comprise a size and complexity of the index.

41. The system of claim 40, wherein the index comprises a binary tree structure, and wherein a height of the index tree is indicative of the size and complexity of the index.

42. The system of claim 38, wherein the means for determining which operation is more efficient further comprises considering at least one of a following factors: an estimated time required to extract index keys from the table; an estimated time to sort the index keys; and an estimated time to rebuild the index from the sorted keys.

43. The system of claim 38, further comprising:

means for maintaining a list of threshold values for different index sizes; and

means for using the number of received data records to add to the table to determine a comparison value, wherein determining whether the first or second operation is more efficient is based on the comparison value and the threshold for the size of the index to be updated.

44. The system of claim 43, wherein the comparison value comprises the number of the received data records as a percentage of all data records in the table.

45. The system of claim 44, wherein the index comprises a binary tree and wherein the list of threshold values provides one threshold for each of a plurality of

different height index binary trees, wherein the threshold selected for comparison with the comparison value is based on the height of the index to update.

46. The system of claim 43, wherein the first operation is more efficient if the comparison value is less than the threshold value and wherein the second operation is more efficient if the comparison value is greater than the threshold value.

47. A program for updating an index on a database table when data is added to the table, wherein the program is embedded in a computer readable medium and capable of causing a computer to perform:

receiving data records to load into the table;

selecting one of a first operation and second operation, wherein the first operation incrementally updates the index on the table as each received data record is added to the table and the second operation rebuilds the index from the table after all the received data records have been added to the table; and

using the first operation or second operation determined to update the index with the received data.

48. The program of claim 47, wherein the program is further capable of causing the processor to perform:

determining which of the first operation or second operation is more efficient, wherein the first or second operation determined to be more efficient is selected to use for updating the index with the received data.

49. The program of claim 48, wherein determining which operation is more efficient is a function of a percentage of the received data records to add to the table and characteristics of the index.

50. The program of claim 49, wherein the characteristics of the index used in determining which operation is more efficient comprise a size and complexity of the index.

51. The program of claim 50, wherein the index comprises a binary tree structure, and wherein a height of the index tree is indicative of the size and complexity of the index.

52. The program of claim 49, wherein determining which operation is more efficient further comprises considering at least one of a following factors: an estimated time required to extract index keys from the table; an estimated time to sort the index keys; and an estimated time to rebuild the index from the sorted keys.

53. The program of claim 49, wherein the program is further capable of causing the processor to perform:

maintaining a list of threshold values for different index sizes; and

using the number of received data records to add to the table to determine a comparison value, wherein determining whether the first or second operation is more efficient is based on the comparison value and the threshold for the size of the index to be updated.

54. The program of claim 53, wherein the comparison value comprises the number of the received data records as a percentage of all data records in the table.

55. The program of claim 54, wherein the index comprises a binary tree and wherein the list of threshold values provides one threshold for each of a plurality of different height index binary trees, wherein the threshold selected for comparison with the comparison value is based on the height of the index to update.

56. The program of claim 53, wherein the first operation is more efficient if the comparison value is less than the threshold value and wherein the second operation is more efficient if the comparison value is greater than the threshold value.

# HEURISTIC-BASED CONDITIONAL DATA INDEXING

## ABSTRACT

A computer system for the indexing of data in which a heuristic determination function is applied to predict an efficient index updating approach. The system is able to update an index relating to a first data set by incrementally updating the index or by a rebuild of the index at the completion of the addition of a second set of data to the first set of data. The system applies a heuristic determination function to the characteristics of the first set of data, its index, and the second set of data, to predict whether an incremental update or a rebuild update of the index will result in a more efficient rebuild of the data. The system applies this approach to a restore and rollforward recovery or a data load operation to improve the efficiency of these operations.

Exp. Mail # EL484106450US

I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. 1.10 on the date indicated below and is addressed to the Box: PATENT APPLICATION, Assistant Commissioner for Patents, Washington, D.C. 20231 on July 28, 2000

Patricia McLaughlin

7/28/00
Dated

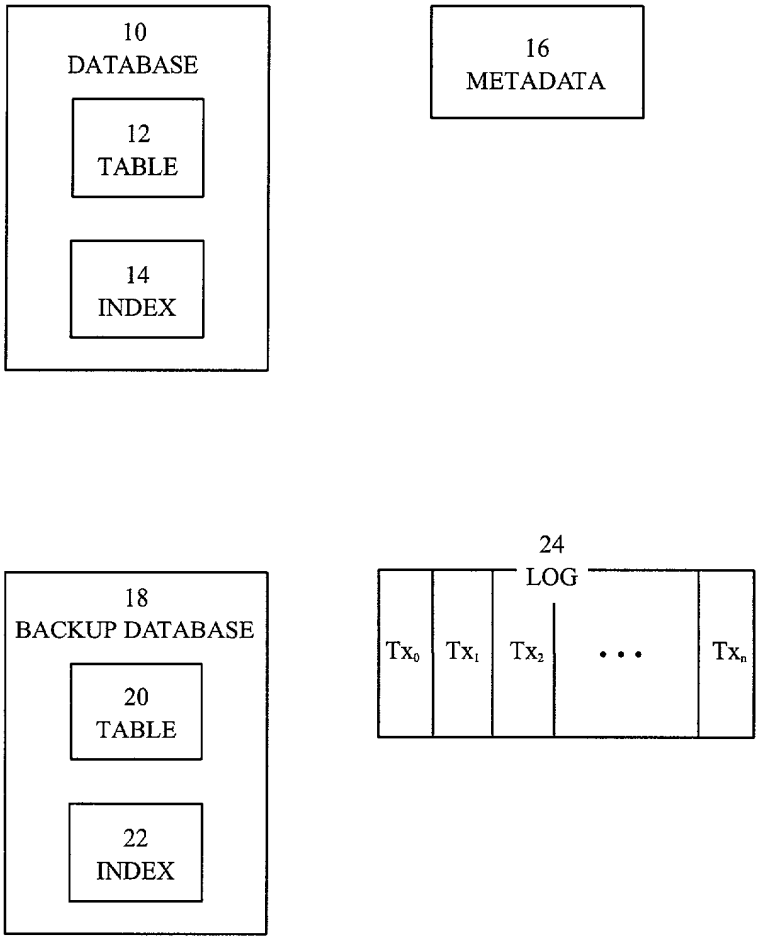
10
DATABASE
12
TABLE
14
INDEX
16
METADATA
18
BACKUP DATABASE
20
TABLE
22
INDEX
24
LOG
$Tx_0$
$Tx_1$
$Tx_2$
• • •
$Tx_n$

**FIGURE 1**

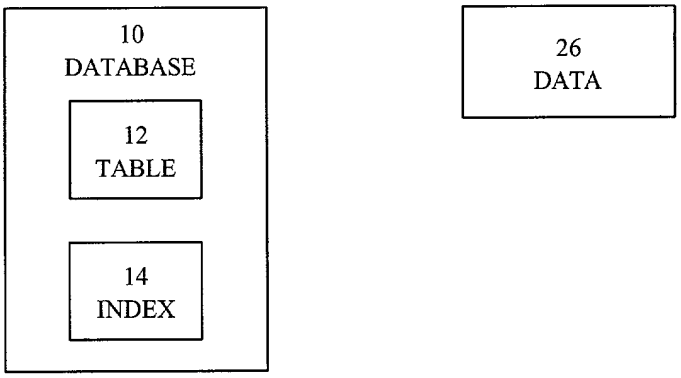


**FIGURE 2**

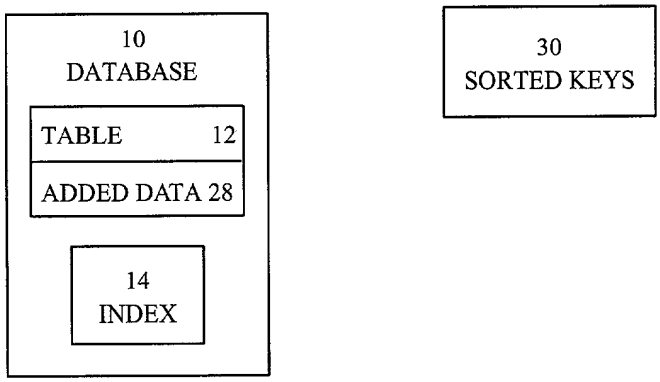


**FIGURE 3**

**DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION**

**DOCKET: CA990022US1**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

METHOD AND SYSTEM FOR IMPROVING CONCURRENCY THROUGH EARLY RELEASE OF UNNECESSARY LOCKS

the specification of which (check one)

X is attached hereto.
___ was filed on _____
as Application Serial No. _____
and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d) or Section 365(b) of any foreign application(s) for patent or inventor's certificate, or Section 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate or PCT International application having a filing date before that of the application on which priority is claimed:

| Prior Foreign Application(s) | | | Priority Claimed |
|---|---|---|---|
| 2,279,119 | Canada | 29/7/99 | x Yes _No |
| (Number) | (Country) | (Day/Month/Year Filed) | |

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) or Section 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56, which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

| | | |
|---|---|---|
| None | | |
| (Application Serial No.) | (Filing Date) | (Status) (patented, pending, abandoned) |

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

**DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION**

**DOCKET: CA990022US1**

**POWER OF ATTORNEY:** As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. (list name and registration number) Romualdas Strimaitis, Reg. No. 35,697; Prentiss W. Johnson, Reg. No. 33,123; Ingrid M. Foerster, Reg. No. 36,511; Timothy M. Farrell, Reg., No. 37,321; Christopher A. Hughes, Reg. No. 26,914; John E. Hoel, Reg. No. 26,279; Edward A. Pennington, Reg. No. 32,588; Joseph C. Redmond, Jr., Reg. No. 18,753; David W. Victor, Reg. No. 39,867; William K. Konrad, Reg. No. 28,868; Alan S. Raynes, Reg. No. 39,809.

Send correspondence to:

**David Victor, Esq**
**1180 South Beverly Dr., Ste. 501**
**Los Angeles, CA 90035**

Direct all telephone calls to David Victor at (310) 553-7977

| **FULL NAME OF INVENTOR ONE: Sam S. Lightstone** | |
|---|---|
| INVENTORS SIGNATURE: | DATE: |
| RESIDENCE: 60 Hillmount Avenue, Toronto, Ontario Canada M4B 1X4 | |
| CITIZENSHIP: Canada | |
| POST OFFICE ADDRESS: same as residence | |

| **FULL NAME OF INVENTOR TWO: Catherine S. McArthur** | |
|---|---|
| INVENTORS SIGNATURE: | DATE: |
| RESIDENCE: 549 Marksbury Road, Pickering, Ontario Canada L1W 2S7 | |
| CITIZENSHIP: Canada | |
| POST OFFICE ADDRESS: same as residence | |

| **FULL NAME OF INVENTOR THREE: William T. O'Connell** | |
|---|---|
| INVENTORS SIGNATURE: | DATE: |
| RESIDENCE: 130 Tilman Circle, Markham, Ontario Canada L3P 6A2 | |
| CITIZENSHIP: United States of America | |
| POST OFFICE ADDRESS: same as residence | |

| FULL NAME OF INVENTOR FOUR: Miroslaw A. Plasza | |
|---|---|
| INVENTORS SIGNATURE: | DATE: |
| RESIDENCE: 2961 Dufferin Street, Apt. 405, Toronto, Ontario Canada M5B 3T2 | |
| CITIZENSHIP: Polish | |
| POST OFFICE ADDRESS: same as residence | |